Error Reconciliation in Quantum Key Distribution Protocols

Miralem Mehic[1,3(✉)], Marcin Niemiec[2,3], Harun Siljak[4], and Miroslav Voznak[3]

[1] Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Zmaja od Bosne bb, Kampus Univerziteta, 71000 Sarajevo, Bosnia and Herzegovina
miralem.mehic@ieee.org
[2] AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Krakow, Poland
[3] Department of Telecommunications, VSB-Technical University of Ostrava, 17. listopadu 15, 70800 Ostrava-Poruba, Czech Republic
[4] CONNECT Centre, Trinity College Dublin, Dunlop Oriel House 34 Westland Row, Dublin 2, Ireland

Abstract. Quantum Key Distribution (QKD) protocols allow the establishment of symmetric cryptographic keys up to a limited distance at limited rates. Due to optical misalignment, noise in quantum detectors, disturbance of the quantum channel or eavesdropping, an error key reconciliation technique is required to eliminate errors. This chapter analyses different key reconciliation techniques with a focus on communication and computing performance. We also briefly describe a new approach to key reconciliation techniques based on artificial neural networks.

Keywords: Error reconciliation · Quantum key distribution · Performances · Reversibility

1 Introduction

QKD provides an effective solution for resolving the cryptographic key establishment problem by relying on the laws of quantum physics. Unlike approaches based on mathematical constraints whose security depends on the attacker's computational and communication resources, QKD does not put a limit on the available resources but limits the length of the link implementation [1]. A QKD link can be realized only to a certain distance and at certain rates since it involves usage of two channels: quantum/optical and public/classical.

This work has been partially supported by COST Action IC1405 on Reversible Computation - Extending Horizons of Computing, and partly by the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 830943, the ECHO project. This work was also supported by the Ministry of Education, Science and Youth of Canton Sarajevo, Bosnia and Herzegovina under Grant No. 11/05-14-27719-1/19 and partly by the Horizon 2020 project OpenQKD under grant agreement No. 857156.

I. Ulidowski et al. (Eds.): RC 2020, LNCS 12070, pp. 222–236, 2020.
https://doi.org/10.1007/978-3-030-47361-7_11

Quantum cryptography focuses on photons (particles of light), using some of their properties to act as an information carrier. Principally, information is encoded in a photon's polarization; a single polarized photon is referred to as a qubit (quantum bit) which cannot be split, copied or amplified without introducing detectable disturbances.

The procedure for establishing a key is defined by QKD protocol, and three basic categories are distinguished: the oldest and widespread group of discrete-variable protocols (BB84, B92, E91, SARG04), efficient continuous-variable (CV-QKD) protocols and distributed-phase-reference coding (COW, DPS) [2,3]. The primary difference between these categories is reflected in the method of preparing and generating photons over a quantum channel [4–6].

A quantum channel is used only to exchange qubits, and it provides the QKD protocol with raw keys. All further communication is performed over a public channel, and it is often denoted as post-processing. It includes steps that need to be implemented for all types of protocols [2], exchanging only the accompanying information that helps in the profiling of raw keys. The overall process is aimed at establishing symmetric keys on both sides of the link in a safe manner.

The initial post-processing step is called a sifting phase, and it is used to detect those qubits for which adequate polarization measurement bases have been used on both sides. Therefore, user B, typically designated Bob informs user A, usually named Alice in literature, about bases he used, and Alice provides feedback advising when incompatible measurement bases have been used. It is important to underline that information about the measurement results is not revealed since only details on used bases are exchanged. Bob will discard bits for cases when incompatible bases have been used, providing the sifted key.

Further, it is necessary to check whether the eavesdropping of communication has been performed. This step is known as error-rate estimation since it is used to estimate the overall communication error. The eavesdropper is not solely responsible for errors in the quantum channel since errors may occur due to imperfection in the state preparation procedure at the source, polarization reference frame misalignment, imperfect polarizing beam splitters, detector dark counts, stray background light, noise in the detectors or disturbance of the quantum channel. However, the threshold of bit error rate p_{max} for the quantum channel without the presence of eavesdropper Eve is known in advance, and this information can be compared with the measured quantum bit error rate (QBER) p of the channel. The usual approach for estimation of the QBER in the channel (p) is to compare a small sample portion of measured values. The selected portion should be sufficient to make the estimated QBER credible where the question about the length of the sample portion is vital [4,7,8]. After estimating QBER, the obtained value can be compared with the already known threshold value of p_{max}. If the error rate is higher than a given threshold ($p > p_{max}$), the presence of Eve is revealed which means that all measured values should be discarded and the process starts from the beginning. Otherwise, the process continues.

Although the estimated value is lower than the threshold value, there are still measurement errors that need to be identified, and those bits need to be corrected or discarded. The process of locating and removing errors is often denoted as "error key reconciliation". As shown in traffic analysis experiments [9, 10], error key reconciliation represents a highly time demanding and extensive computational part of the whole process. Depending on the implementation, a key reconciliation step may affect the quantum channel and considerably impact the key generation rate.

In the following sections, we analyze the most popular error reconciliation approaches. Cascade protocol is discussed in Sect. 2, overview of Winnow protocol is given in Sect. 3. Section 4 outlines LDPC approach while the comparison is given in Sect. 5. We introduce the new key reconciliation protocol in Sect. 6 and provide conclusion in Sect. 7.

2 Cascade

The most widely used error key reconciliation protocol is cascade protocol due to its simplicity and efficiency [11]. Cascade is based on iterations where random permutations are performed with the aim of evenly dispersing errors throughout the sifted key. The permuted sifted key is divided into equal blocks of k_i bits, and after each iteration and new permutations, the block size is doubled: $k_i = 2 \cdot k_{i-1}$. The results of the parity test for each block are compared, and a binary search to find and correct errors in the block is performed. However, to improve the efficiency of the process, the cascade protocol investigates errors in pairs of iterations in a recursive way.

Instead of rejecting error bits in the first stage, information about the presence of an error bit in the block is used in the further iterations to detect errors that have not been detected due to the measurement parity. For any error detected in further iterations, at least one matching error can be identified in the same block of the previous iteration which was previously considered as a block without errors. Using a binary search, a deep search for errors in such a block is performed, and the masked errors can be recursively detected. Two passes of cascade protocol are illustrated in Fig. 1.

The length of the initial block k_1 is a critical parameter which depends on the estimated QBER. The empirical analysis described in [11] proposes the use of value $k_1 = 0.73/p$ as the optimal value, where p is the estimated QBER. Sugimoto modified the cascade protocol to bring the cascading protocol closer to theoretical limits [12]. Besides, he confirmed that four iterations are sufficient for the effective key reconciliation as originally proposed in [11]. However, due to the dependence of the initial block's length on the estimated QBER, it is advisable to execute all the iterations (as long as the length of the block k_i is not equal to the length of the key). In [4], Rass and Kollmitzer showed that adopting block-size to variations of the local error rate is worthwhile, as the efficiency of error correction can be increased by reducing the number of bits revealed to an adversary [13].

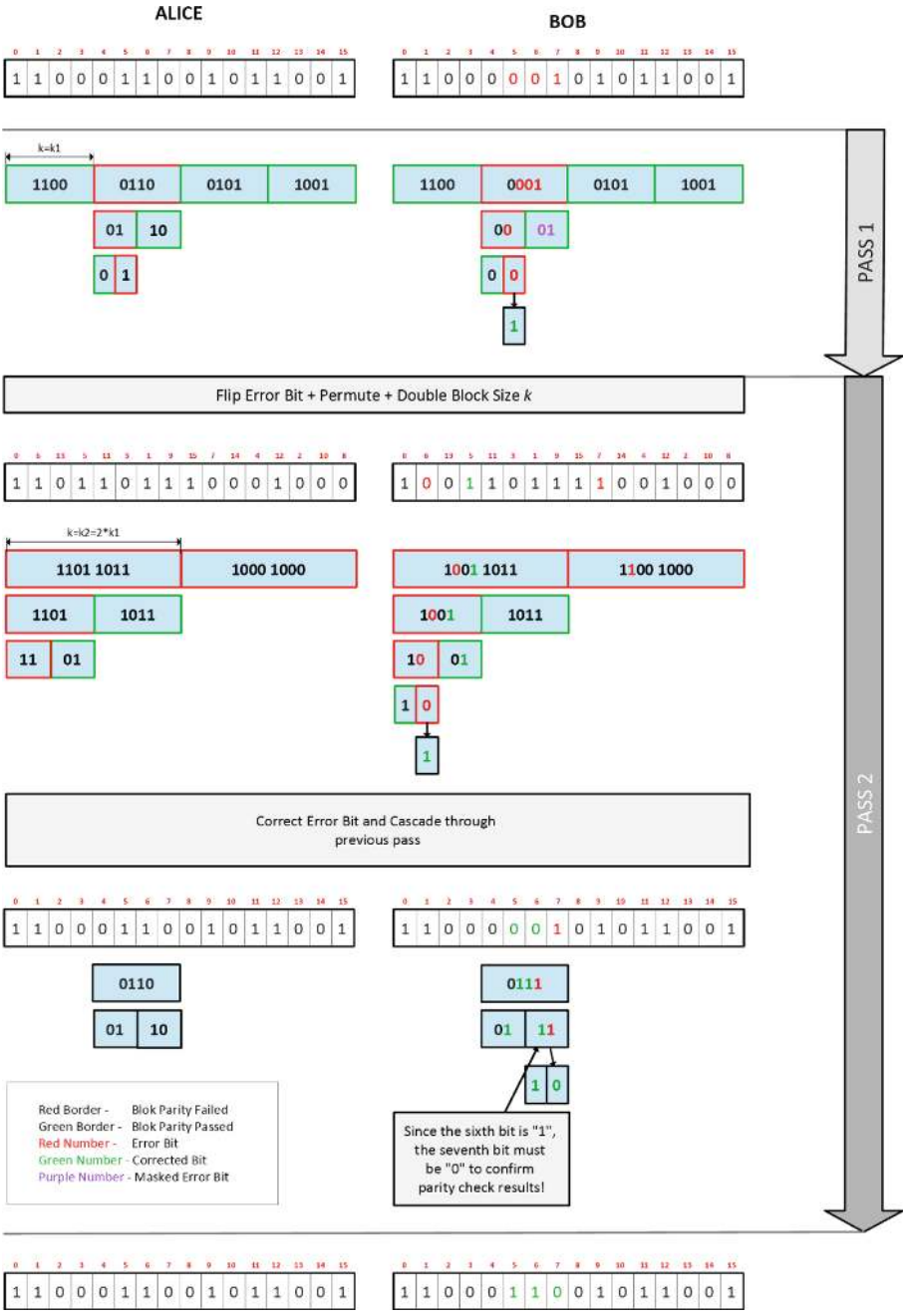

Fig. 1. Illustration of the first two passes of reconciliation using a Cascade protocol.

Cascade protocol relies on the use of the binary search to locate an error bit. The binary search includes further division of the block into two smaller subblocks for which the results of parity check values are compared until an error is found. For each block with an error bit, in total $1 + \lceil \log_2 k_i \rceil$ parity values are exchanged since $1 + \lceil \log_2 k_i \rceil$ is the maximum number of times that block k_i can be splitted, and only one parity value is exchanged for blocks without errors.

In addition to discarding the sample portion bits used to estimate QBER value, it is advised to discard the last bit of each block and subblock for which the parity bit was exchanged to minimize the amount of information gained by Eve. The maximum number of discared bits denoted as D_i can be calculated based on k_i value in the i^{th} iteration as follows:

$$\sum D_i = \sum_i \Big(\sum_{\substack{initially \\ even \\ blocks}} 1 + \sum_{\substack{initially \\ odd \\ blocks}} (1 + \lceil log_2\ k_i \rceil) + \sum_{\substack{other \\ errors \\ corrected}} \lceil log_2\ k_i \rceil \Big) \quad (1)$$

As proposed in [14], Eq. (1) can be shortened to:

$$D = \sum D_i = \sum_i \Big(\frac{n}{k_i} + \sum_{\substack{errors \\ corrected}} \lceil log_2\ k_i \rceil \Big) \quad (2)$$

where $k_i = 2 \cdot k_{i-1}, k_i < \frac{n}{2}$ and n denotes the amount of the measured values in sifting phase. The number of discarded bits depends on the QBER value and initial block size. However, Sugimoto showed [12] that most errors are corrected in the first two iterations. The empirical analysis of cascade protocol is given in [15], while the practical impact of cascade protocols on post-processing is considered in [9,16]. In [17], Chen proposed the extension of random permutations using interleaving technique optimized to reduce or eliminate error clusters from burst errors. Nguyen proposed modifying the permutation method used in cascade [18]. Yan and Martinez proposed modifications based the initial key's length in [19,20] while the use of Forward Error Correction was analyzed in [21] (Table 1).

Table 1. Error correction per passes using Cascade protocol

Iteration	1	2	3	4
Corrected errors (%)	54.522%	45.347%	0.451%	0.002%

3 Winnow

In 2003, Winnow protocol based on Hamming codes was introduced [22]. The aim was to increase the throughput and reduce the interactivity of Cascade by eliminating the binary search step.

Both parties, Alice and Bob, divide their random keys M_a and M_b into blocks of equal length (recommended starting size is $k = 8$) and calculate syndrome values S_a and S_b based on a Generator matrix G and a parity check Matrix H where $H \cdot G^T = 0$. For each block of size k, based on his key values M_b, Bob will generate and transmit his syndrome $S_b = H \cdot M_b$ to Alice, which will calculate the syndrome differences S_d. If S_d is non-zero, Alice will attempt to correct the errors with the fewest changes leading to syndrome zero values.

$$S_a = H \cdot M_a^T = \begin{bmatrix} 0\,1\,1\,0\,1\,0\,1 \\ 1\,0\,1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} 0\,1\,1\,0\,0\,1\,1 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$S_b = H \cdot M_b^T = \begin{bmatrix} 0\,1\,1\,0\,1\,0\,1 \\ 1\,0\,1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} 0\,1\,0\,0\,0\,1\,1 \end{bmatrix}^T = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$S_a \bigotimes S_b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 = 3 \ (\textit{bit on position } 3 \textit{ is the error}) \tag{3}$$

The Hamming distance d_{min} between codewords limits the number of errors that are suitable for correction where a code word with the number of errors greater than $\frac{d_{min}}{2}$ may closely resemble different code word then correcting the considered code word. Due to reliance on Hamming codes, the Winnow protocol may actually introduce errors, which is the main disadvantage of the shortly described approached. Its efficiency is lower when compared to Cascade for QBER values below 10% that are useful for practical QKD [23].

To achieve information-theoretical secrecy, Buttler suggested discarding an additional bit of each block of size k in the privacy maintenance phase [22].

4 Low Density Parity Check

With terrestrial links, Alice and Bob are usually not limited to execution time, computation and communication complexity. However, with satellite links, the parties need to consider significant losses in the channel, limited time to establish a key due to periodic satellite passage where communication and computation complexity puts additional constraint. Therefore, in previous years, researchers have turning to the application of Gallager's Low Density Parity Check (LDPC) codes that have recently been shown to reconcile errors at rates higher than those of Cascade and Winnow [24–26]. LDPC provides low communication overhead and inherent asymmetry in the amount of computation power required at each side of the channel.

LDPC linear codes are based on a parity check matrix H and a generator matrix G where a decoding limit of the code is defined with the minimum distance. The dimensions of H and G are $m \times n$ where $m = n \cdot (1-r)$ and r is defined

as code rate in range [0, 1]. The code rate value is usually defined beforehand; it defines the correcting power and efficiency. The reconciliation algorithm based on LDPC includes following steps:

- An estimation of QBER of the communication channel is performed,
- Based on estimated QBER, Alice and Bob choose the same $m \times n$ generator matrix G and parity check matrix H,
- For each sifted key, Bob calculates syndrome S_b and send it to Alice,
- Alice attempts to reconcile sifted key, assuming that Bob has the correct sifted key. Her goal is to resolve Bob's key vector x, based on her key vector y, received syndrome S_b, the parity-check matrix H, and estimated QBER value. Alice can use several techniques to decode LDPC such as belief propagation decoding algorithm (also known as the Sum-Product algorithm) or Log-Likelihood Ratios which significantly lower computational complexity [4,16,23].

Decoding LDPC code requires larger computational and memory requirements than either the Cascade or Winnow algorithms. However, it has a significant advantage due to the reduction of communication resources since only one information exchange is required. In networks with limited resources (bandwidth and latency), such tradeoff provides potentially large gains in overall runtime and secrecy. In the context of QKD, LDPC was firstly used as a base for the BBN Niagara protocol in DARPA QKD network [27].

5 Comparisson

For testing purposes, Cascade, Winnow and LDPC code were implemented in C++ programming language on servers Intel (R) Xeon (R) Silver 4116 CPU @ 2.10 GHz with 8 GB, and 512 GB HDD. For each value of QBER, 10.000 random keys were tested with the same random seed, which allowed repeating scenarios for different protocols used (Cascade, Winnow and LDPC). In total, 870,000 tests were performed.

The total number of leaked bits is defined as follows:

- Cascade: For each exchange of parity value, one bit is discarded.
- Winnow: For each block k, one bit is discarded.
- LDPC: Total length of syndrome S_b value exchanged.

Figure 2 shows that for small values of QBER (up to 0.05%), Cascade quickly finds and removes errors resulting in a small number of iterations. However, as the QBER value increases (up to 0.10%), LDPC shows better efficiency in terms of overhead and information exchanged.

Figure 3 shows that the overhead efficiency has its price in terms of execution time. Due to the simplicity of algorithms, Cascade and Winnow codes have almost fixed execution time, while in LDPC, the code execution time varies, and gradually increases with the QBER increase.

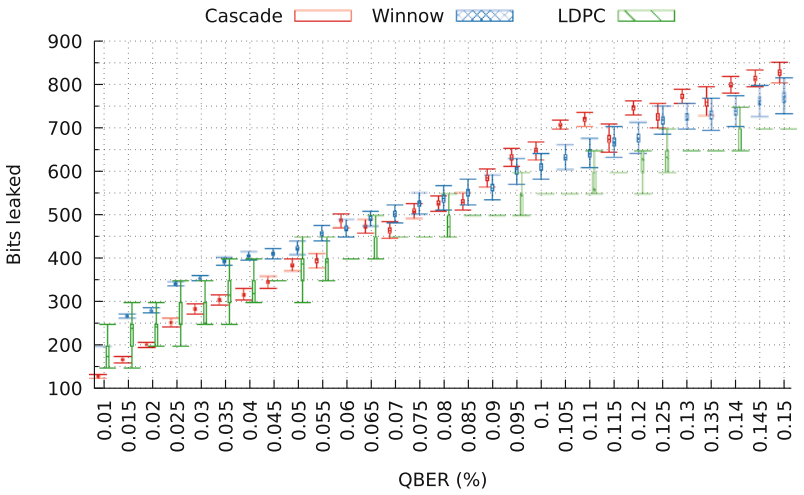

Fig. 2. The number of bits leaked (discarded) for different QBER values. Due to its simplicity, the binary search within Cascade protocol can locate errors in a short time for lower values of QBER. However, for more significant QBER values, binary search requires deeper checking of the sifted key, which increases communication. In the case of Winnow, syndrome message per each block of length k is exchanged which can be used to detect errors in early stages.

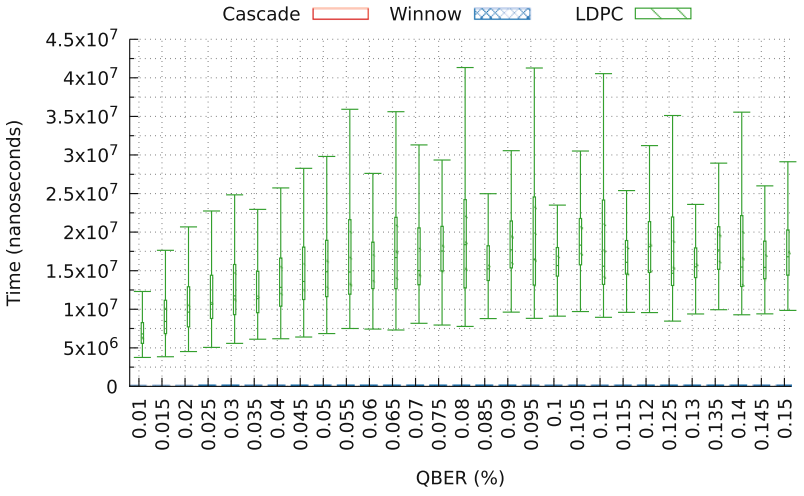

Fig. 3. The execution time for different QBER values. LDPC predominantly requires more time to execute key reconciliation tasks while due to its simplicity, the execution time of the Cascade and Winnow protocols is almost constant. LDPC based on the belief propagation algorithm was used for decoding.

6 Error Correction Based on Artificial Neural Networks

Using artificial neural networks for error correction during a key reconciliation process is a new concept, introduced in [28]. This proposal assumes the use of mutual synchronization of artificial neural networks to correct errors occurring during transmission in the quantum channel. Alice and Bob create their own neural networks based on their keys (with errors). After the mutual learning process, they correct all errors and can use the final key for cryptography purposes.

6.1 Tree Parity Machines

Tree parity machine (TPM) is a type of artificial neural networks (ANN) – a family of statistical learning models inspired by biological neural networks [29]. It consists of artificial neurons (analogous to biological neurons) which are connected and are able to transmit a signal from one neuron to another [30]. Neurons are usually organized in layers: the first layer consists of input neurons which can send the data to the second layer (called hidden). The last layer – called the output layer – consists of output neurons. TPM contains only one hidden layer and has a single neuron in the output layer. It consists of KN input neurons, where K is the number of neurons in the hidden layer and N is the number of inputs into each neuron in the hidden layer. An example of TPM is presented in Fig. 4.

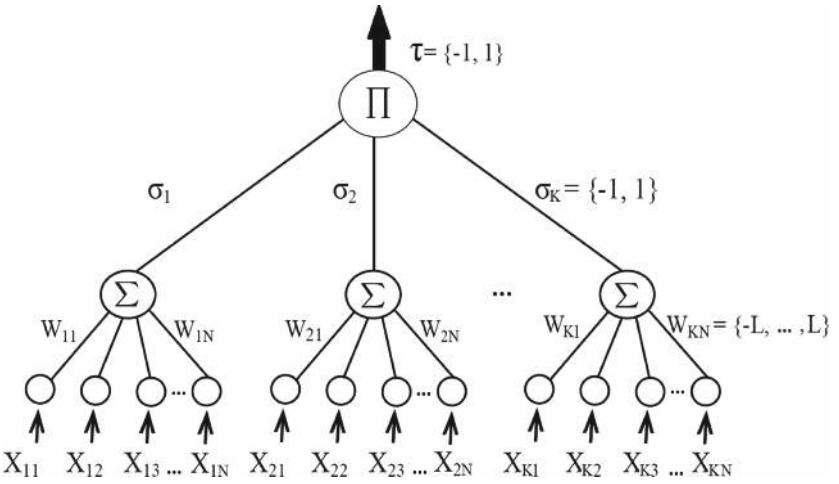

Fig. 4. Structure of TPM machine [28]

TPMs have another important feature: connections between neurons can store parameters (called weights) that can be manipulated during calculation.

Each connection between the input layer and hidden layer is characterized by its weight, which is an integer from the range $[-L, L]$. The output value of neuron k in the hidden layer depends on input x and weight w and is calculated as:

$$\sigma_k = sgn(\sum_{n=1}^{N} x_{kn} * w_{kn}) \tag{4}$$

where signum function is:

$$sgn(z) = \begin{cases} -1 & z \leq 0 \\ 1 & z > 0 \end{cases} \tag{5}$$

The output value of the neuron in the output layer is calculated as:

$$\tau = \prod_{k=1}^{K} \sigma_k \tag{6}$$

When Alice and Bob build their own TPMs with the same structure (K, N and L), they can synchronize these artificial networks after mutual learning [31]. At the beginning of this process, each TPM generates random values of weights, however after the synchronization process both users have TPMs with the same values of weights. Therefore, Alice and Bob can use this phenomenon to correct errors occurring in the quantum channel.

In order to synchronize neural networks, Alice or Bob generates random inputs and both users compute outputs from each TPM. If the outputs have the same value, they start the learning process, but if the outputs are different, a new string of bits must be generated. Alice and Bob can choose any learning algorithm; however, the generalized form of Hebbian method is the most popular in practical implementations [32]. This algorithm strengthens the connections which have the same value as the TPM output. The new weights are calculated by means of the following formula:

$$w_{kn}^{\star} = \nu_L(w_{kn} + x_{kn} * \sigma_k * \Theta(\sigma_k, \tau)) \tag{7}$$

where:

$$\Theta(\sigma_k, \tau)) = \begin{cases} 0 & \text{if } \sigma_k \neq \tau \\ 1 & \text{if } \sigma_k = \tau \end{cases} \tag{8}$$

and function ν_L limits values of connections to the range $[-L, L]$:

$$\nu_L(z) = \begin{cases} -L & \text{if } z \leq -L \\ z & \text{if } -L < z < L \\ L & \text{if } z \geq L \end{cases} \tag{9}$$

After the appropriate number of iterations, the synchronization process ends, and the weights of both TPM machines are the same. However, synchronization

of TPMs requires public channel for communication between Alice and Bob where Eve can eavesdrop and try to synchronize her own TPM machine with Alice and Bob. Fortunately, if the output of Eve's TPM machine is different than the outputs of Alice and Bob's machines, the learning process cannot be performed. Therefore, the synchronization of Eve's TPM is much slower than the synchronization of the TPMs belonging to Alice and Bob. An example of the synchronization process is presented in Fig. 5 (TPM machines with parameters: N = 8, K = 6, L = 2 and Hebbian learning algorithm). Alice and Bob synchronized neural networks before 200 iterations, but the attacker was not able to do it for 1000 iterations.

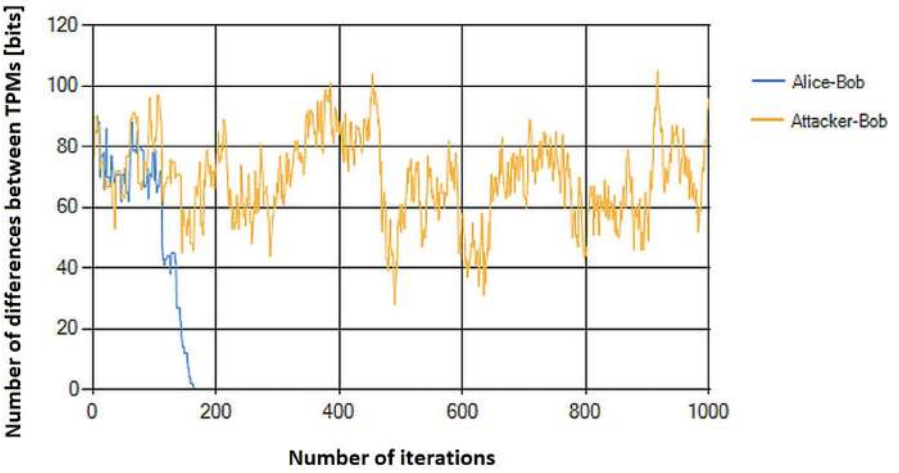

Fig. 5. Example of TPMs synchronization: Alice's TPM and Bob's TPM, Bob's TPM and Attacker's TPM (TPM machines with parameters: N = 8, K = 6, L = 2 and Hebbian learning algorithm)

6.2 Error Correction Based on TPMs

We can use the presented synchronization of the TPM machines to correct errors in the quantum cryptography. In the beginning, Alice and Bob create their own TPM machines based on their own strings of bits. The users change the string of bits into weights in their own TPM machines (bits into numbers from the range $[-L, L]$). Values $\{-L, -L+1, ...L-1, L\}$ become weights of connections between the input neurons and the neurons in the hidden layer. In this way, Alice and Bob construct very similar neural networks – the TPM machines have the same structure, and most of the weights are the same. The differences are located only in the places where errors occurred: for example, if QBER ≈3%, it means that ≈97% of bits are correct. After this, synchronization of the

TPM machines begins and continues until all weights in both machines become the same. When each random input is chosen (input strings have KN length), the users compute outputs and compare the obtained values. When the TPM machines are synchronized, the weights are the same in both neural networks. Therefore Alice and Bob can convert the weights back into bits because both strings are now the same. All errors have been corrected.

Importantly, Alice's binary string is very similar to Bob's string of bits. The typical value for QBER does not exceed a few percent; therefore we must correct only a small part of the whole key. This means that the TPM machines are close to synchronization and the learning process will finish much faster than in the case of synchronization of random strings of bits. Of course, this increases the security level significantly.

It is worth mentioning that this idea – using the mutual synchronization of neural networks to correct errors – is a special case when this process makes sense. In general, TPM machines cannot be used for error correction of digital information because we are not able to predict the final weights after the learning process.

7 Conclusion

In this chapter, we analyzed techniques of implementing the key reconciliation using Cascade, Winnow, Low-density parity-check code and the application of neural networks with a focus on communication and computing performances.

Our previous results [9] showed that key reconciliation process takes the dominant part of QKD post-processing. With increasing interest in satellite and global QKD connections, minimizing the duration of key establishment process is becoming an increasingly attractive area. It is necessary to take into account the possibilities of asymmetric processing, which simplifies the requirements for computing power budgets as well as requirements for minimizing the exchange of packets to reduce overhead and the ability to work in networks with weaker network performance (bandwidth and network delay).

Since the development of metropolitan QKD testbed networks [33–39], LDPC is increasingly being considered as an adequate basis for the key reconciliation process in QKD, and there are noticeable variations in how this protocol is implemented. However, techniques of reversibility or on artificial neural networks can significantly improve the process to reduce communication and computing resources and represent areas of great interest for further research.

References

1. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, New York, vol. 175, p. 8 (1984)
2. Scarani, V., Bechmann-Pasquinucci, H., Cerf, N.J., Dušek, M., Lütkenhaus, N., Peev, M.: The security of practical quantum key distribution. Rev. Mod. Phys. **81**(3), 1301–1350 (2009)

3. Assche, G.V.: Quantum Cryptography and Secret-Key Distribution. Cambridge University Press, Cambridge (2006)
4. Kollmitzer, C., Pivk, M.: Applied Quantum Cryptography, vol. 1. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-04831-9
5. Dodson, D., et al.: Updating quantum cryptography report ver. 1. arXiv preprint arXiv:0905.4325, May 2009
6. Dusek, M., Lutkenhaus, N., Hendrych, M.: Quantum cryptography. In: Progress in Optics, vol. 49, pp. 381–454. Elsevier, January 2006
7. Niemiec, M., Pach, A.R.: The measure of security in quantum cryptography. In: 2012 IEEE Global Communications Conference (GLOBECOM), pp. 967–972, December 2012
8. Mehic, M., Niemiec, M., Voznak, M.: Calculation of the key length for quantum key distribution. Elektron. Elektrotech. **21**(6), 81–85 (2015)
9. Mehic, M., Maurhart, O., Rass, S., Komosny, D., Rezac, F., Voznak, M.: Analysis of the public channel of quantum key distribution link. IEEE J. Quantum Electron. **53**(5), 1–8 (2017)
10. Mehic, M., et al.: A novel approach to quality-of-service provisioning in trusted relay quantum key distribution networks. IEEE/ACM Trans. Netw. **28**(1), 168–181 (2020)
11. Brassard, Gilles, Salvail, Louis: Secret-key reconciliation by public discussion. In: Helleseth, Tor (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 410–423. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_35
12. Sugimoto, T., Yamazaki, K.: A study on secret key reconciliation protocol. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E83-A**(10), 1987–1991 (2000)
13. Lustic, K.: Performance analysis and optimization of the winnow secret key reconciliation protocol. Ph.D. thesis, Air Force Institute of Technology (2010)
14. Ruth, Y.: A probabilistic analysis of binary and cascade. math.uchicago.edu (2013)
15. Calver, T.: An empirical analysis of the cascade secret key reconciliation protocol for quantum key distribution. Master thesis (2011)
16. Pedersen, T.B., Toyran, M., Pearson, D., Pedersen, T.B., Toyran, M.: High performance information reconciliation for QKD with CASCADE. Quantum Inf. Comput. **734**(5–6), 419–434 (2013)
17. Keath, C.: Improvement of reconciliation for quantum key distribution. Ph.D. thesis, Rochester Institute of Technology, February 2010
18. Nguyen, K.C.: Extension des protocoles de réconciliation en cryptographie quantique. Université Libre de Bruxelles, Travail de fon d'études (2002)
19. Yan, H., et al.: Information reconciliation protocol in quantum key distribution system. In: Proceedings - 4th International Conference on Natural Computation, ICNC 2008, vol. 3, pp. 637–641 (2008)
20. Martinez-Mateo, J., Pacher, C., Peev, M., Ciurana, A., Martin, V.: Demystifying the information reconciliation protocol cascade. arXiv preprint arXiv:1407.3257, pp. 1–30, July 2014
21. Nakassis, A., Bienfang, J.C., Williams, C.J.: Expeditious reconciliation for practical quantum key distribution. In: Donkor, E., Pirich, A.R., Brandt, H.E. (eds.) Quantum Information and Computation II, vol. 5436, p. 28, August 2004
22. Buttler, W.T., Lamoreaux, S.K., Torgerson, J.R., Nickel, G.H., Donahue, C.H., Peterson, C.G.: Fast, efficient error reconciliation for quantum cryptography. Phys. Rev. A **67**(5), 052303 (2003)
23. Elkouss, D., Leverrier, A., Alleaume, R., Boutros, J.J.: Efficient reconciliation protocol for discrete-variable quantum key distribution, June 2009

24. Gallager, R.G.: Low-density parity-check codes. IRE Trans. Inf. Theory **8**, 21–28 (1962)
25. Elkouss, D., Martinez-Mateo, J., Vicente, M.: Information reconciliation for QKD. Quantum Inf. Comput. **11**(March), 226–238 (2011)
26. Elkouss, D., Martinez-Mateo, J., Martin, V.: Analysis of a rate-adaptive reconciliation protocol and the effect of leakage on the secret key rate. Phys. Rev. A - At. Mol. Opt. Phys. **87**(4), 1–7 (2013)
27. Elliott, C., Colvin, A., Pearson, D., Pikalo, O., Schlafer, J., Yeh, H.: Current status of the DARPA quantum network (Invited Paper). In: Donkor, E.J., Pirich, A.R., Brandt, H.E. (eds.) Quantum Information and Computation III. Proceedings of SPIE, vol. 5815, pp. 138–149, May 2005
28. Niemiec, M.: Error correction in quantum cryptography based on artificial neural networks. Quantum Inf. Process. **18**(6), 174 (2019)
29. Kanter, I., Kinzel, W.: The theory of neural networks and cryptography (2007)
30. Hadke, P.P., Kale, S.G.: Use of neural networks in cryptography: a review. In: IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (2016)
31. Chakraborty, S., Dalal, J., Sarkar, B., Mukherjee, D.: Neural synchronization based secret key exchange over public channels: a survey. In: 2014 International Conference on Signal Propagation and Computer Technology, ICSPCT 2014 (2014)
32. Kriesel, D.: A brief introduction on neural networks. Technical report, December 2007. www.dkriesel.com
33. Elliott, C., Yeh, H.: DARPA quantum network testbed. Technical report July, BBN Technologies Cambridge, New York, USA, New York (2007)
34. Alleaume, R., et al.: SECOQC white paper on quantum key distribution and cryptography. arXiv preprint quant-ph/0701168, p. 28 (2007)
35. Korzh, B., et al.: Provably secure and practical quantum key distribution over 307 km of optical fibre. Nat. Photon. **9**(3), 163–168 (2015)
36. Sasaki, M.: Tokyo QKD network and the evolution to secure photonic network. In: CLEO:2011 - Laser Applications to Photonic Applications, vol. 1, JTuC1. OSA, Washington, D.C. (2011)
37. Dixon, A.R., Yuan, Z.L., Dynes, J.F., Sharpe, A.W., Shields, A.J.: Continuous operation of high bit rate quantum key distribution. Appl. Phys. Lett. **96**(2010), 2008–2011 (2010)
38. Salvail, L., Peev, M., Diamanti, E., Alléaume, R., Lütkenhaus, N., Länger, T.: Security of trusted repeater quantum key distribution networks. J. Comput. Secur. **18**(1), 61–87 (2010)
39. Shimizu, K., et al.: Performance of long-distance quantum key distribution over 90-km optical links installed in a field environment of Tokyo metropolitan area. J. Lightwave Technol. **32**(1), 141–151 (2014)